In the Claims:

Please amend claims 1, 6, 12 and 17 as indicated below.

1. (Currently amended) A system, comprising:

one or more host machines configured to implement a plurality of instances of an application server; and

one or more client machines each configured to implement one or more clients of the application server, wherein each client on a respective one of the one or more client machines is configured to:

create a ~~plurality of~~ different client-side Object Request Broker (ORB) ~~Brokers (ORBs)~~ on the client machine for each of the plurality of application server instances to generate a plurality of client-side ORBs each corresponding to a different one of the application server instances, wherein each client-side ORB is coupled to a server-side ORB of ~~a different one of the plurality of~~ its corresponding application server ~~instances~~ instance;

select one of the plurality of client-side ORBs created by the client on the client machine according to a load balancing scheme in response to a request to access the application server; wherein the load balancing scheme comprises a random scheme, a round robin scheme, or an intelligent scheme based on feedback; wherein the load balancing scheme distributes requests to access the application server from the client among the plurality of client-side ORBs created by the client, thereby distributing the requests to access the application server from the client among the plurality of application server instances; and

access ~~a particular one of~~ the ~~plurality of~~ application server ~~instances~~ instance corresponding to the selected client-side ORB via the selected client-side ORB coupled to [[a]] the server-side ORB of the ~~particular~~ corresponding application server instance.

2. (Previously presented) The system as recited in claim 1, wherein said access of a particular one of the plurality of application server instances via the selected client-side ORB is performed according to RMI-IIOP (Remote Method Invocation – Internet Inter-ORB Protocol).

3. (Original) The system as recited in claim 1, wherein said creation of a plurality of client-side ORBs and said selection of one of the plurality of client-side ORBs according to a load balancing scheme are performed by a Context Factory class.

4. (Previously presented) The system as recited in claim 3, wherein the Context Factory class is a factory class of a naming and directory interface that provides naming and directory functionality to applications.

5. (Previously presented) The system as recited in claim 1, wherein each client on a respective one of the one or more client machines is further configured to:

select a different one of the plurality of client-side ORBs on the client machine according to the load balancing scheme in response to another request to access the application server; and

access a different one of the plurality of application server instances using the different client-side ORB coupled to a server-side ORB of the different application server instance.

6. (Currently amended) A client machine, comprising:

a processor; and

a memory comprising program instructions, wherein the program instructions are executable by the processor to implement:

create a plurality of client-side Object Request Brokers (ORBs) on the client machine for a client of an application server, wherein a different client-side ORB is created on the client machine for each of a plurality of instances of the application server to generate a plurality of client-side ORBs each corresponding to a different one of the application server instances, wherein the client is on the client machine, wherein each client-side ORB is coupled to a server-side ORB of ~~a different one of the plurality of instances~~ its corresponding instance of the application server on one or more host machines;

select one of the plurality of client-side ORBs on the client machine according to a load balancing scheme in response to a request to access the application server; wherein the load balancing scheme comprises a random scheme, a round robin scheme, or an intelligent scheme based on feedback; wherein the load balancing scheme distributes requests to access the application server from the client among the plurality of client-side ORBs, thereby distributing the requests to access the application server from the client among the plurality of application server instances; and

access ~~a particular one of~~ the ~~plurality of~~ application server ~~instances~~ instance corresponding to the selected client-side ORB via the selected client-side ORB coupled to [[a]] the server-side ORB of the ~~particular~~ corresponding application server instance.

7. (Previously presented) The client machine as recited in claim 6, wherein said access of a particular one of the plurality of application server instances via the selected client-side ORB is performed according to RMI-IIOP (Remote Method Invocation – Internet Inter-ORB Protocol).

8. (Previously presented) The client machine as recited in claim 6, wherein said creation of a plurality of client-side ORBs and said selection of one of the plurality of client-side ORBs according to a load balancing scheme are performed by a Context Factory class.

9. (Previously presented) The client machine as recited in claim 8, wherein the Context Factory class is a factory class of a naming and directory interface that provides naming and directory functionality to applications.

10. (Previously presented) The client machine as recited in claim 6, wherein the program instructions are further executable by the processor to:

select a different one of the plurality of client-side ORBs on the client machine according to the load balancing scheme in response to another request to access the application server; and

access a different one of the plurality of application server instances using the different client-side ORB coupled to a server-side ORB of the different application server instance.

11. (Canceled)

12. (Currently amended) A computer-implemented method, comprising:

creating a plurality of client-side Object Request Brokers (ORBs) on a client

machine for a client application on the client machine, wherein a different client-side ORB is created on the client machine for each of a plurality of instances of an application server to generate a plurality of client-side ORBs each corresponding to a different one of the application server instances, wherein each client-side ORB is coupled to a server-side ORB of ~~a different one of a plurality of instances of an~~ its corresponding application server instance;

the client application requesting access to the application server;

selecting one of the plurality of client-side ORBs on the client machine according to a load balancing scheme in response to the request; wherein the load balancing scheme comprises a random scheme, a round robin scheme, or an intelligent scheme based on feedback; wherein the load balancing scheme distributes requests to access the application server from the client application among the plurality of client-side ORBs, thereby distributing the requests to access the application server from the client application among the plurality of application server instances; and

the client application accessing ~~a particular one of~~ the ~~plurality of~~ application server ~~instances~~ instance corresponding to the selected client-side ORB via the selected client-side ORB coupled to [[a]] the server-side ORB of the ~~particular~~ corresponding application server instance.

13. (Previously presented) The computer-implemented method as recited in claim 12, wherein said accessing a particular one of the plurality of application server instances via the selected client-side ORB is performed according to RMI-IIOP (Remote Method Invocation – Internet Inter-ORB Protocol).

14. (Previously presented) The computer-implemented method as recited in claim 12, wherein said creating a plurality of client-side ORBs and said selecting one of

the plurality of client-side ORBs according to a load balancing scheme in response to the request are performed by a Context Factory class.

15. (Previously presented) The computer-implemented method as recited in claim 14, wherein the Context Factory class is a factory class of a naming and directory interface that provides naming and directory functionality to applications written in Java programming language.

16. (Previously presented) The computer-implemented method as recited in claim 12, further comprising:

the client application requesting another access to the application server;

selecting a different one of the plurality of client-side ORBs on the client machine according to the load balancing scheme in response to the other request; and

the client application accessing a different one of the plurality of application server instances using the different client-side ORB coupled to a server-side ORB of the different application server instance.

17. (Currently amended) A computer-accessible storage medium comprising program instructions, wherein the program instructions are computer-executable to implement:

creating a plurality of client-side Object Request Brokers (ORBs) on a client machine for a client application the client machine, wherein a different client-side ORB is created on the client machine for each of a plurality of instances of the application server to generate a plurality of client-side ORBs each corresponding to a different one of the application server instances, wherein each client-side ORB is coupled to a server-side ORB

of ~~a different one of a plurality of instances of an~~ its corresponding application server instance;

receiving a request from the client application for access to the application server;

selecting one of the plurality of client-side ORBs on the client machine according to a load balancing scheme in response to the request; wherein the load balancing scheme comprises a random scheme, a round robin scheme, or an intelligent scheme based on feedback; wherein the load balancing scheme distributes requests to access the application server from the client application among the plurality of client-side ORBs, thereby distributing the requests to access the application server from the client application among the plurality of application server instances; and

the client application accessing ~~a particular one of~~ the ~~plurality of~~ application server ~~instances~~ instance corresponding to the selected client-side ORB via the selected client-side ORB coupled to [[a]] the server-side ORB of the ~~particular~~ corresponding application server instance.

18. (Previously presented) The computer-accessible storage medium as recited in claim 17, wherein said accessing a particular one of the plurality of application server instances via the selected client-side ORB is performed according to RMI-IIOP (Remote Method Invocation – Internet Inter-ORB Protocol).

19. (Previously presented) The computer-accessible storage medium as recited in claim 17, wherein said creating a plurality of client-side ORBs and said selecting one of the plurality of client-side ORBs according to a load balancing scheme in response to the request are performed by a Context Factory class.

20. (Previously presented) The computer-accessible storage medium as recited in claim 19, wherein the Context Factory class is a factory class of a naming and directory

interface that provides naming and directory functionality to applications.

21. (Previously presented) The computer-accessible storage medium as recited in claim 17, wherein the program instructions are further computer-executable to implement:

receiving another request from the client application for access to the application server;

selecting a different one of the plurality of client-side ORBs on the client machine according to the load balancing scheme in response to the other request; and

the client application accessing a different one of the plurality of application server instances using the different client-side ORB coupled to a server-side ORB of the different application server instance.